Title
=====
DANA, Distributed Asynchronous Numerical & Adaptive computing framework

Motivations
===========

Computational neuroscience is a vast domain of research going down from the very precise modeling of a single spiking neuron, taking into account ion channels and/or dendrites spatial geometry up to the modeling of very large assemblies of simplified neurons that are able to give account of complex cognitive functions. DANA attempts to address this latter modeling activity by offering a python computing framework for the design of large assemblies of neurons using numerical and distributed computations.

Implementation
==============

DANA is a python computing framework based on numpy and scipy libraries whose primary goals relate to computational neuroscience and artificial neural networks. However, this framework can be used in several different areas like physic simulations, cellular automata or image processing. The computational paradigm supporting the DANA framework is grounded on the notion of a unit that is a essentially a set of arbitrary values that can vary along time under the influence of other units and learning. Each unit can be connected to any other unit (including itself) using a weighted (possibly adaptive) connection. A group is a structured set of such homogeneous units.

For example the game of life can be written very simply as follow:

```
>>> from dana import *
>>> src = Group((100,100),
          '''V = maximum(0,1.0-(N<1.5)-(N>3.5)-(N<2.5)*(1-V)) : int
             N : float''')
>>> C = SparseConnection(src('V'), src('N'),
                         np.array([[1., 1., 1.],
                                   [1., 0., 1.],
                                   [1., 1., 1.]]))
```

src has been defined as a group of 100x100 units, each of them having a single value V whose value is computed according to V equation and where N designates a connection from Z to Z (using specified kernel in C definition). To simulate the game of life, one can write:

```
>>> src.V = rnd.randint(0, 2, src.shape)
>>> run(n=100)
```

In the example above, the connection has been made static, but it could have been made adaptive by defining an equation dW/dt for the C connection.

The DANA framework offers a set of core objects to design and run such models with the possibilty of specifying model equations (differentials/regular), connections and connection equations.

Finally, it is to be noted that DANA has been made purposely very similar to the BRIAN spiking neural networks simulator and attempt to re-use its syntax as much as possible.

More information at http://dana.loria.fr